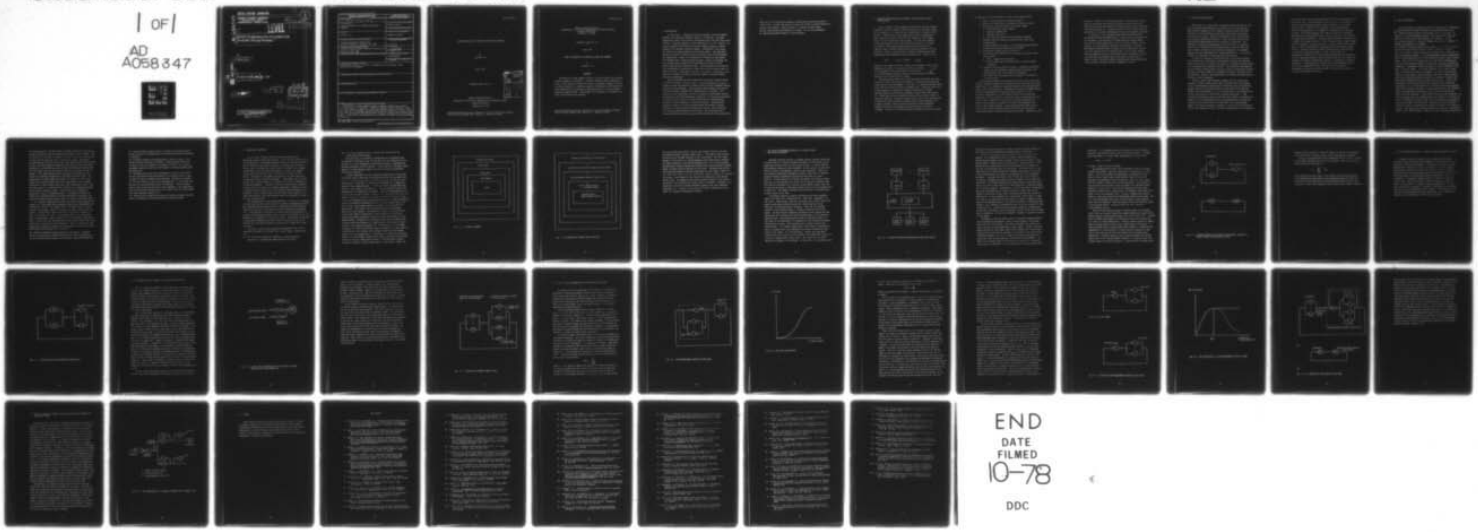
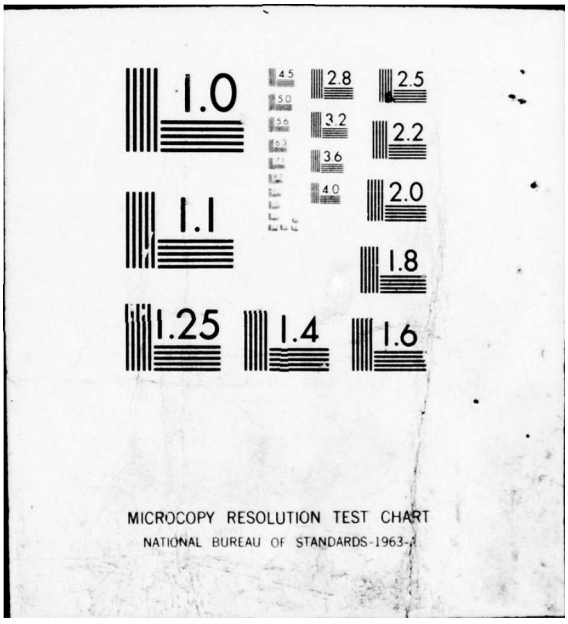
AD-A058 347 STANFORD UNIV CALIF STANFORD ELECTRONICS LABS F/G 9/2
NOTES ON MODELLING OF COMPUTER SYSTEMS AND NETWORKS.(U)
APR 78 P S YU DASG60-77-C-0073
UNCLASSIFIED SU-SEL-78-016 NL

1 OF 1
AD A058 347



END
DATE FILMED
10-78
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

DIGITAL SYSTEMS LABORATORY

STANFORD ELECTRONICS LABORATORIES
DEPARTMENT OF ELECTRICAL ENGINEERING
STANFORD UNIVERSITY · STANFORD, CA 94305

SU-SEL-78-016
DSL-TR-154

NOTES ON MODELLING OF COMPUTER SYSTEMS AND NETWORKS.

by

Philip S. Yu

Technical Report No. 154

April 1978

49 p.

The work described herein was supported in part by the Ballistic Missile Defense Systems Command under contract no. DASG60-77-C-0073.

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: SU-SEL 78-016, DSL Tech. Rept. # 154	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE *(and Subtitle)*: NOTES ON MODELLING OF COMPUTER SYSTEMS AND NETWORKS	5. TYPE OF REPORT & PERIOD COVERED: Technical Report
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Philip S. Yu	8. CONTRACT OR GRANT NUMBER(s): DASG60-77-C-0073
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Stanford Electronics Laboratories, Stanford University, Stanford, CA 94305	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 6.33.04.A
11. CONTROLLING OFFICE NAME AND ADDRESS: Ballistic Missile Defense Advanced Technology Ctr., ATC-P, P.O. Box 1500, Huntsville, AL 35807	12. REPORT DATE: April 1978
	13. NUMBER OF PAGES: 45
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*	15. SECURITY CLASS. *(of this report)*: Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Reproduction in whole or in part is permitted for any purpose of the United States Government.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

Formulation of given computer system or network problems into abstract stochastic models is considered. Generally speaking, model formulation is an art. While analytic results are clearly not powerful enough to provide a "cook-book" approach to modelling, general methodology and difficulties on model formulation are discussed through examinations of various computer system and network models. These models are presented in a systematic way based on the hierarchical approach.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

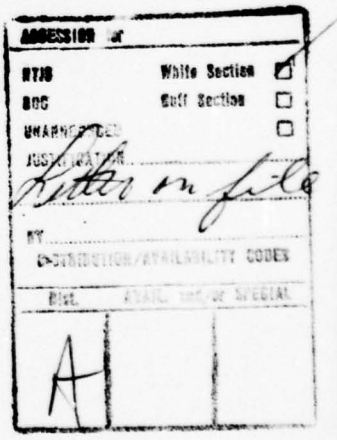
SU-SEL 78-016

NOTES ON MODELLING OF COMPUTER SYSTEMS AND NETWORKS

by

Philip S. Yu

April 1978



Technical Report No. 154

Digital Systems Laboratory

Departments of Electrical Engineering and Computer Science

Stanford University

Stanford, CA 94305

The work described herein was supported in part by the Ballistic Missile Defense Systems Command under contract no. DASG60-77-C-0073.

SU-SEL 78-016

Digital Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, CA 94305

Technical Report No. 154

April 1978

NOTES ON MODELLING OF COMPUTER SYSTEMS AND NETWORKS

by

Philip S. Yu

ABSTRACT

Formulation of given computer system or network problems into abstract stochastic models is considered. Generally speaking, model formulation is an art. While analytic results are clearly not powerful enough to provide a "cookbook" approach to modelling, general methodology and difficulties on model formulation are discussed through examinations of various computer system and network models. These models are presented in a systematic way based on the hierarchical approach.

The work described herein was supported in part by the Ballistic Missile Defense Systems Command under contract no. DASG60-77-C-0073.

I INTRODUCTION

In this report, formulation of given computer system or network problems into abstract stochastic models is considered. The model formulation task is very crucial and sometimes more difficult than model solving. Generally speaking, model formulation is an art. There do not exist systematic rules which can convert computer system or network problems into abstract stochastic models. Furthermore, the stochastic model for a given problem is by no means unique. The same problem can be modelled to various levels of details under various degrees of complexities. Since it is impossible to include all the details of a problem into a single stochastic model, we have to determine the most crucial factors affecting the performance and try to incorporate them into the stochastic model. If the problem can be modelled by a conventional queueing model, there often exist systematic ways to obtain an exact or approximate solution of the model. Otherwise, we can only try to attack the problem using more general stochastic models and ad hoc techniques.

We will consider the model formulation of typical computer system or network problems in fairly details. A multiprogrammed time shared computer system may consist of several loosely coupled computers sharing secondary storage devices. Each loosely coupled computer may consist of several tightly coupled CPU's contending for main memory. The computer system itself may be part of a computer communication network. An important modelling methodology referred to as hierarchical modelling is introduced and used as the basic foundation for systematically modelling various problems in a bottom up fashion. Problems being modelled consist of the interference problem and software lockout problem in a tightly coupled multiprocessor system, similar problems in a loosely coupled multiprocessor system, the performance of a multiprogrammed time shared computer system and finally the transmission

delay in a store and forward computer communication network via terrestrial links. We hope the reader can gain some feeling on model formulation through these models. There usually is a tradeoff between mathematical tractability and fidelity of a model. We are just presenting or comparing possible models of a given problem.

2. QUEUEING NETWORK MODELS WITH PRODUCT FORM EQUILIBRIUM STATE POSSIBILITIES

A job or process in the computer system usually will go through a sequence of service requests on different devices at different stages of its lifetime. To be more precise, starting with a request for CPU service, a process will later request service from a paging device when a page fault occurs. After service completion at the paging device, it will request CPU service again. The same story repeats. It may also request service from I/O devices later on. To capture this kind of correlations among different service devices in a computer system, queueing network models have been proven to be very effective. There is a broad class of queueing network models whose equilibrium state probabilities are in product form. That is to say

$$P(S_1, \ldots, S_M) = G\, P(S_1) \ldots P(S_M)$$

where M is the number of service centers in the network. $P(S_i)$ is the probability that the i-th queue is in state S_i and $P(S_1, \ldots, S_M)$ is the probability that the network is in state $(S_1, \ldots, S_M)$. G is a normalization constant, chosen so that the sum of the probabilities of all network states is unity.

Queueing network models were first introduced by Jackson [42] where the service time at each server has negative exponential distribution. A good survey on the development of the theory of queueing network models up to 1972 can be found in Muntz and Baskett [61]. The recent most noteworthy progress in extending the class of analytically solvable queueing networks has been done by Baskett, Chandy, Muntz and Palacios [4]. These authors have succeeded in casting into a unified theory of previously known but unconnected results such as queue size distributions for M/M/1 with FCFS discipline, general service time distribution for processor sharing, infinite server discipline and LCFS preemptive resumed discipline and queueing systems with various classes of customers. For ease of illustrating model formulation in later sections,

we summarize the characteristics of these network models below.

1. There is no restriction on network topology.
2. There is no restriction on the number of service centers.
3. Four kinds of service centers are allowed, namely
   a. First come first served (FCFS)
   b. Processor sharing (PS)
   c. Infinite server (IS)
   d. Last come first served (LCFS) preemptive resumed.
4. Customers traverse through the network under some fixed routing chains.
5. A routing chain may be open or closed depending upon whether external arrivals and departures are allowed.
6. The external arrival process is Poisson.
7. The service time distribution at each type of service center is as follows
   a. FCFS: exponential distribution
   b. PS, IS, LCFS: any distribution with a rational Laplace transform.
8. Customers can have different classes. Customers in different classes can have different routing chains and service time distributions at PS, IS or LCFS service centers.
9. The service rate may depend upon the number of customers in a service center.

Reiser and Kobayashi [67] generalize the above result to the case in which customer transitions are characterized by more than one closed Markov chain. The technique of generating function has been applied to obtain closed form solutions. Kobayashi and Reiser [54] further extend the job routing behavior to high order Markov chains, i.e. the transition probability of a job from one station to the other can depend on, at least, the last two stations it has visited and not just the last one. In Lam [55], the class of queueing networks with a product form solution is extended to include state dependent lost arrivals and trigger arrivals. Such queueing network models can be used to model store and forward packet switching nodes and multi-programming computer systems with storage constraint. Chandy et al [16]

introduces a property of queueing discipline, station balance, which seems to explain why some service disciplines yield product form solutions for queues and networks with nonexponential service time and other disciplines do not. Previous results on local balance are also generalized to arbitrary differentiable service distribution functions.

In addition to the stationary queue length distribution and performance measures derivable from it, such as device utilization and job throughput, distributions of response times or passage times, which are the times required for a job to traverse a portion of the network, are other measures receiving a great amount of interest. Nevertheless, analyses based on number-in-queue state space can only yield the expected values of passage times. Yu [8] obtain passage time distributions for the closed network models in Baskett et al [4] by transforming the passage time problem into a hitting time of a Markov process. Efficient numerical approximation and generalization to open queueing networks are also considered in Yu [8].

3. DIFFUSION APPROXIMATION

There still is a large class of queueing systems which do not fit into the queueing network models described in the previous section. One particular example will be a general queueing network with first come first served discipline and general service time distribution at each service center. Recently, considerable effort has been made for obtaining approximate solutions to these queueing systems using the diffusion approximation when the systems are congested. The advantage of diffusion approximation lies in the fact that explicit results can be obtained for relatively complex situations where the only possible alternatives are simulation experiments or sometimes numerical methods (polynomial root finding methods) [4]. This greatly extends our capability in modelling practical problems.

In order to alleviate the difficulty due to the general service time distribution, the diffusion approximation replaces the discrete jump process such as the queue size process by a diffusion process which is a continuous path stochastic process. The probability distribution of the diffusion process which satisfies a partial differential equation is quite often more amenable to mathematical analysis than that of the jump process. However, the approximation by diffusion process requires the heavy traffic assumption.

Based on central limit theorem, Kingman [46] has shown in his treatment of heavy traffic theory that the waiting time distribution is as an approximation exponentially distributed, where the parameter depends only on the mean and variance of the interarrival time and service time distribution, i.e., it is insensitive to the detailed form of the distribution, as the traffic intensity approaches 1. The diffusion approximation based on the same idea attempts to overcome the limitation of the exponential model by considering both the mean and variance of the service time and interarrival time distributions. Newell [62] gives an extensive treatment of queues with time dependent arrival rate through use of the diffusion approximation in his monograph.

Gaver applies the diffusion approximation method to the waiting time in a M/G/1 queue [30]. Gaver and Shedler [32][33] apply this technique to analyze a multiprogrammed computer system modelled as a two stage cyclic network. Kobayashi [50] considers the multi-dimensional diffusion approximation as a technique for treating general queueing networks. Reiser and Kobayashi [67] study the accuracy of diffusion approximation techniques and propose a way to treat each server in the queueing networks separately. Gleenbe [34][35] suggests a different way to handle the boundary condition of the diffusion process, namely using Feller's elementary return process [24]. In [36], Gelenbe also investigates the idea of decomposing a queueing network into separate single servers. Halachni and Franta [38][39] extend diffusion approximation to multiserver queueing systems. Yu [80] investigates the applicability conditions of diffusion approximation and the decomposability of queueing networks, which have been neglected in the past, and improves its accuracy by employing a new way to estimate the diffusion parameters. Generalization to servers with a queue dependent service rate is also considered in [80]. An application of diffusion approximation to analyze the performance of an ALOHA-like system can be found in Kobayashi et al [53].

4. MODEL FORMULATION

In this section, we summarize some basic guidelines for formulating a computer system and network problems into queueing models below:

(1) Identify service centers and customers in a given problem. For example, in a multiprogrammed computer system model, CPU, drums and disks are service centers and processes are customers. In the communication subnet of a computer communication network, channels are servers and messages are customers.

(2) Identify the type of each service center. In the Markovian queueing network models described by Baskett, Chandy, Muntz and Palacios [6], four kinds of service centers are allowed, namely, FCFS, PS, IS and LCFS premptive resumed. In a multiprogrammed computer system model, CPU is usually modelled as PS or LCFS premptive resumed type of service center where PS is the limiting case of round robin scheduling and LCFS premptive resumed represents an efficient preemptive resumed CPU scheduling. Furthermore, drums and disks are usually modelled as FCFS type of service centers. To be more specific, let us take a closer look on how drums and disks are modelled. As a first order approximation, we often model a multiple spindle disk drive, which consists of several moving head disks connected to a single I/O channel, under FCFS spindle scheduling [79] by a FCFS service center. Recall that accessing a record on moving head disks requires three sequential operations, i.e., the seek operation, rotational positioning operation and the data transfer operation. In the seek operation, the read/write heads are mechanically positioned to the cylinder containing the record that was requested. In the rotational positioning operation, the records passing under the read/write head are examined to find the one to be transferred; the disk is said to be latent during this operation and the time taken is latency time. Finally, in the data transfer operation, data is transmitted through the I/O channel to or from main memory. Each spindle of a multiple spindle disk drive can seek independently of the other spindles. However, the seek commands, which initiate

the seek operations, and the transfer commands, which start the latency-transfer sequences, must pass through the channel to the spindles. The I/O channel can only pass a command or perform a rotational positioning or data transfer operation for one channel at a time. Apparently, the FCFS service center is only a very rough approximation of the actual system. The main problem on modelling comes from the fact that three operations are involved where some of them need dedicated service from the I/O channel and some of them only need to be initiated by the I/O channel. More sophisticated models capturing this problem and the effect of scheduling disciplines can be found in Wilhelm [79] . For paging drums, we often model each sector of a drum as a FCFS service center when shortest-latency-time-first scheduling is used or model each drum as a FCFS service center when FCFS scheduling is used. Note that the access of a record on drums consists of a rotational latency and the actual transfer of data. The latency can be reduced by appropriate choice of scheduling policy. To capture the effect of latency and scheduling policy on system performance, again more complicated stochastic models must be used (see Fuller and Baskett [29]).

In a computer communication network via random access channel, the server, i.e., the random access channel, behaves very differently from conventional servers. For example, under the ALOHA scheme, when the customer (the message) arrives it receives immediate service from the server. If the service time of more than one customer overlaps in time, the service is in vain. All customers have to be served again after a random delay. This is referred to as the message conflict problem. In this case, the conventional queueing network model will not be sufficient to model the system performance. Ad hoc techniques or more sophisticated stochastic models must be employed to evaluate the system performance (see Kleinrock [49] , Yu [82])

(3) Identify different customer classes in the system. Different classes of customers can have different service time distributions at IS, PS and LCFS service centers, and different routing probabilities.

In a multiprogrammed computer system, usually the CPU bound and I/O bound processes are distinguished as two different classes of processes or customers.

(4) Determine whether the queueing model is open or closed. i.e., whether external arrivals are allowed. Closed queueing models will be chosen if the total number of customers is fixed, e.g., a multiprogrammed computer system with fixed degree of multiprogramming under saturated load.

(5) Determine the service time distribution of each service center and the interarrival time distribution if the queueing model is open by empirical data or hardware specifications. Exponential distribution is often used as a first order approximation. In this case, only the mean of the distribution needs to be measured. Notice the mean service time or service rate may be queue dependent. The conventional G/M/m (multiserver) queueing system is a special kind of service center with queue dependent service rate.

(6) Determine the routing chain (hence the network topology) for each class of customers by its characteristic and empirical data.

5. HIERARCHICAL MODELLING

Let us further consider some difficulties encountered in modelling computer systems and networks. [51] The most apparent one is the multiplicity of interacting parameters which are related to activities of different time scales from nano- or microseconds to minutes. Another source of difficulty is that tasks or processors in a computer system often occupy more than one resource at a time. For example, an executing task holds not only a CPU but also a portion of main memory. The amount of main memory allocated to the task clearly has an important effect on system performance. Furthermore, the very same subject can be viewed either as a customer or as a server depending on the problem addressed. A problem of this sort which we will encounter in later sections is that in the memory interference problem the roles of processors are servers, but in the software lockout problem they are customers. Since a subject cannot be both server and customer in the same queueing model, these two problems cannot be formulated in terms of a single queueing model.

The multiplicity of interacting parameters or the size of a computer system or network model can be reduced by properly decomposing the model into submodels and evaluating them in a hierarchical fashion starting from the bottom level if the system is nearly completely decomposable [20]. This approach is referred to as hierarchical modelling which is proposed and formulated by Sekino [71], Courtois [20], Browne et al [11], Brandwajn [9], Kobayashi [51], Brown et al[10]. Furthermore, the concept of hierarchical modelling provides a partial solution to some of the difficulties cited above besides size reduction of a model as we shall see later.

A system is called nearly completely decomposable [20], if all variables can somehow be clustered into a small number of groups such that

(a) The interactions among the variables of each group may be studies as if interactions among groups did not exist.

(b) Interactions among groups be studied with reference to the interactions within groups.

Alternatively, if intergroup dependencies are sufficiently weak as compared to intragroup ones, in the short run the system may be considered as a set of independent subsystems which may be approximately analyzed separately from one another. In the long run, the whole system appears to evolve keeping roughly the state of relative equilibrium within each subsystem.

A basic requirement for near complete decomposability as observed by Courtois [20] is that the intragroup interaction has transient time constants which are much shorter than the mean time between intergroup interactions. To be more concrete, let us consider how to apply the concept of near complete decomposability to computer system modelling. Since the time scale of the interaction plays an essential role in the decomposition, it should not be surprising that the modelling hierarchy and storage hierarchy related. A diagram of a storage hierarchy is presented in Fig. 5.1 and that of an abstract modelling hierarchy based on that in Kobayashi [51] is presented in Fig. 5.2. In this representation of hierarchical modelling, model A is at the lowest level. The time between events being modelled in model A is on the order of a microsecond. The highest level storage involved in the storage hierarchy is main memory. An example of a system at this level will be one with tightly coupled CPU's, each of which has a cache and occasionally makes read/write requests to large main memory or interleaved memory. Problems to be analyzed at this level of models will include memory interference of a multiprocessor, the effect of interleaving, cache mapping algorithm, etc. Model B is one level higher than model A and the time between events in model B is on the order of a millisecond. The highest level storage involved in the storage hierarchy is the paging device. An example of a system at this level will be one with loosely coupled CPU's, each of which has its own main memory and shares common paging devices. Problems such as memory management should be studied using models at this level. Model C is

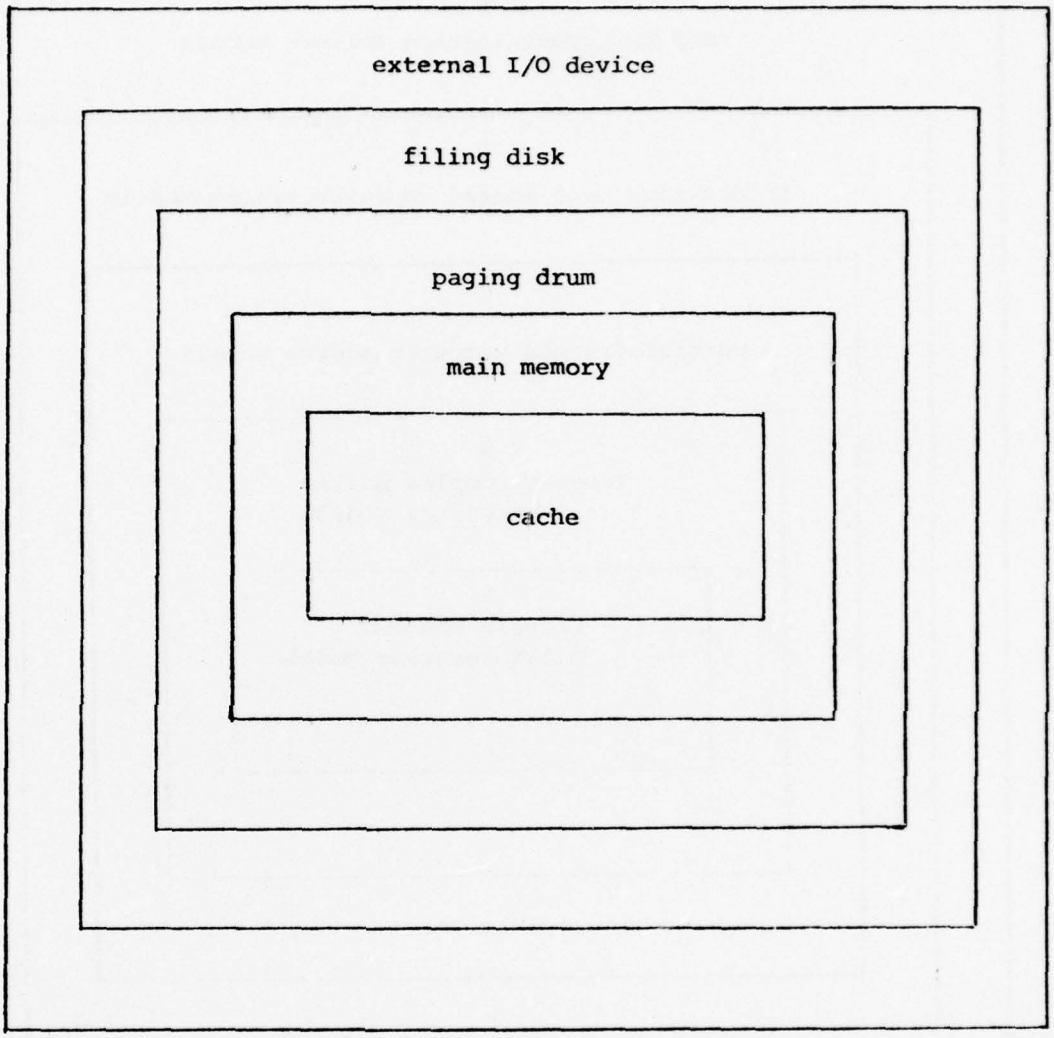


Fig. 5.1 A STORAGE HIERARCHY

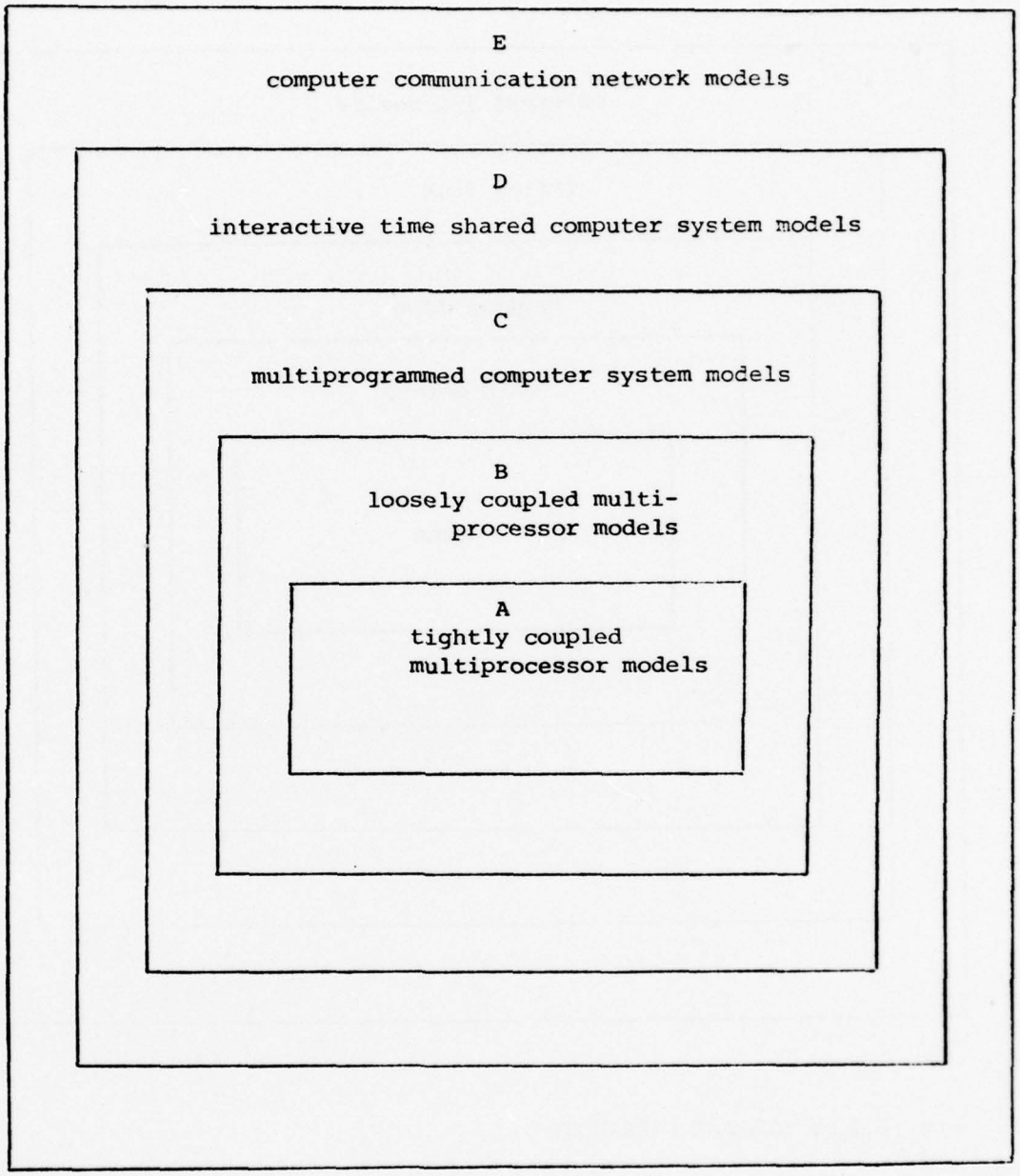


Fig. 5.2 A HIERARCHICAL EXAMPLE MODEL STRUCTURE

one level higher than model B and the time between events is an order of magnitude larger than that under model B. The highest level storage involved in the storage hierarchy is the filing disk. The most common example at this level is a multiprogrammed system model where customers are tasks currently in the multiprogramming mix. If a loosely coupled multiprocessor system communicates through filing disks instead of paging drums, it should also be modelled at this level. Problems such as file I/O scheduling should be studied using models at this level. Model D is one level above model C and the time between events in model C is expressed in seconds. One example at this level is an interactive time sharing system model. The highest level storage device involved in the storage hierarchy is the external I/O devices, such as terminals. Problems to be modelled at this level include terminal response time, job scheduling algorithm, etc. Finally, on the top of the level D model will be the model of a computer communication network.

6. THE MEMORY INTERFERENCE PROBLEM IN A TIGHTLY COUPLED MULTIPROCESSOR SYSTEM

Starting from this section, we examine various computer system and network models in a bottom up fashion according to their levels in the modelling hierarchy cited in the previous section. A tightly coupled multiprocessor system is first examined. In this system, the main store which is usually interleaved is shared by all processors. There are two important performance problems receiving a great deal of interest, namely the memory interference problem and the software lockout problem. In this section, we consider the memory interference problem. Since any of the processors may access any of the memory modules, when more than one processor try to access the same memory module, only one of them can get access to the memory module and the other processors will be queued up. This memory contention has the effect of stretching the average memory access time.

Performance analysis of memory interference has been studied by Skinner and Asher [74] , Bhandarkar [8] , Chow [18] , Kobayashi [51] and others. Nevertheless, when each processor has a private cache, the memory interference problem becomes more complicated due to the difficulty in maintaining data integrity. Tang [75] has recently proposed a cache system design of a tightly coupled multiprocessor system. We first briefly summarize the design of this system and then demonstrate how to convert the design specifications into an abstract queueing model. The multiprocessor system is displayed in Fig. 6.1. The effect of I/O channels will be neglected in the modelling of memory interference as usual. Each processor has a private cache in this configuration. In a typical uniprocessor cache design, a directory is used to translate main store addresses into cache locations for data blocks in the cache. In a multiprocessor system, the same cache organization can be used for each processor, with additional controls to facilitate communications among caches. The "store only in cache" algorithm is adopted in this design. It means that if a processor wants

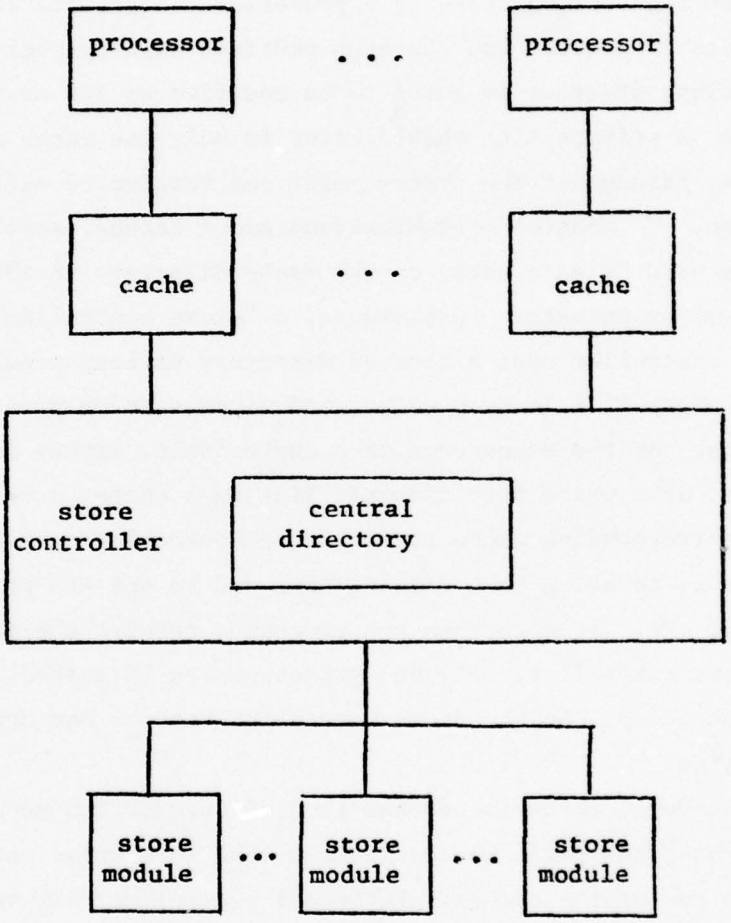


Fig. 6.1 A TIGHTLY COUPLED MULTIPROCESSOR SYSTEM WITH CACHES

to write and a miss occurs in the cache, the data is always brought to the cache so that the processor can always write to the cache.

A "line" is blocks that are associated with a given main store address. In a uniprocessor cache, a line is a block, whereas in a multiprocessor system, a line can exist in more than one cache. A line can be either a shared line or a private line. A shared line is a line which exists at caches and has not yet been modified with respect to its copy at the backing store by a processor. A private line is a line which exists in a cache and has been modified with respect to its copy at the backing store or is going to be modified by its corresponding processor. A private line should exist in only one cache so that at any moment, throughout the system, only one version of data exists for any address. To control communications among caches, status bits would have to be used in each entry of the cache directory to identify whether it is shared or private. Furthermore, a "store controller" is necessary. The store controller uses a central directory to keep track of the status of every line in each cache, and makes sure only one version of data exists. At the occurrence of a cache fault, either due to a read or a write, or a write into a shared line in a cache, a request is sent from the corresponding cache to the store controller to take appropriate actions, e.g. to bring in a missing line and to set the status of a line to private, etc. If more than one processor require the intervention of the store controller, only one processor can be served, and the others will be queued up. So the store controller becomes the critical resource of the system.

Furthermore, there is another kind of interaction among processors which degrades the performance. When a line is granted the status private in one of the caches, the store controller will scratch the other copies of the line appearing in other caches. Clearly, each copy being scratched is very likely still in the working set of the corresponding process and will be requested to send back to the cache shortly afterwards through a cache fault. Hence the interaction increases the cache fault rate of each processor and degrades the performance. Let $S(n)$ be the scratched rate at each cache when there are n processors concurrently in execution, i.e., not pending for service at the central

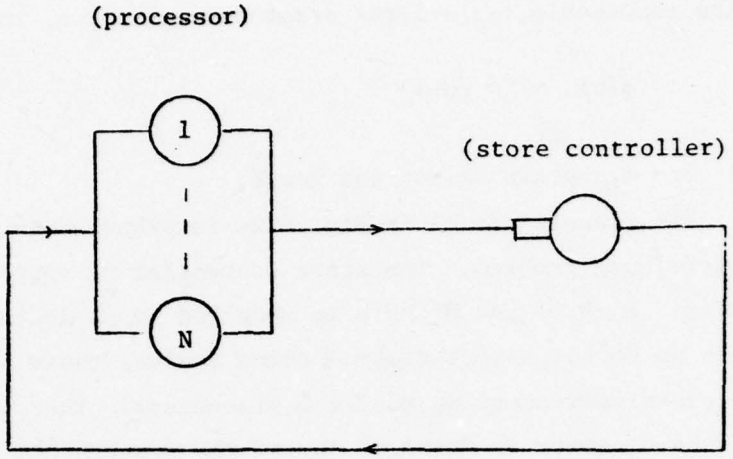
controller. It is apparent that as the number of active processors increases, the scratched rate at each cache will increase. It seems quite reasonable, as a first order approximation, to assume that

$$s(n) = c\,(n-1)$$

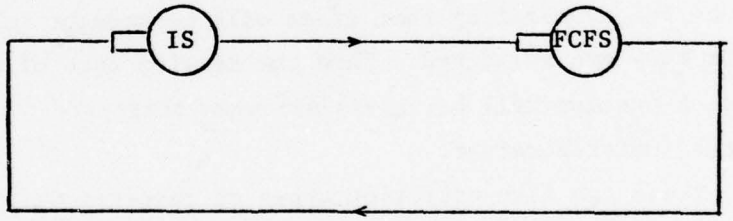
for some c, when n is not too large.

The queueing model in Fig. 6.2a is proposed to model the memory interference problem. The store controller is represented as a FCFS server. Each of the N CPU's is modelled by an independent server. Since we do not switch tasks at cache faults, there are exactly N processes corresponding to the N processors. That is to say, there will be no queue in front of any of the processors. So we should in fact represent the processor service in terms of an infinite server station as in Fig. 6.2.b. Let μ be the request rate to the store controller either due to a cache fault or a change of line status when there is only one processor active. Clearly μ depends upon program behavior, cache size and the line replacement policy of the cache. Let d be the probability that lines will be brought in again shortly after they are scratched. Then the service rate of the infinite server to each process will be $(\mu+ds(n))$ when there are n processes in the infinite server.

There are five different types of requests which can be issued from a cache to the store controller, namely shared read, private read, declare private, replace private line, and replace shared line. A shared read or private read request is issued when a cache fault occurs due to a read or a write, respectively. A declare private request is issued when a processor wants to write into a shared line in its cache. A replace private line or replace shared line request is issued when the replacement of a private line or a shared line is necessary to make room in the cache for the new line coming from the backing store, respectively. Certain requests, such as declare private and replace shared line, do not involve memory transfers. A replace private line request requires a write into memory. A shared read or a private read



(a)



(b)

Fig. 6.2 A QUEUEING MODEL FOR THE MEMORY INTERFERENCE PROBLEM OF A TIGHTLY COUPLED MULTIPROCESSOR SYSTEM

request normally requires a read from memory, but if the line requested is a private line in another cache, a write into memory is required.

Let T_i be the mean service time required by the i-th type of requests at the store controller and α_i be its percentage. Then the mean service time of the store controller is

$$T = \sum_{i=1}^{5} \alpha_i T_i$$

If we assume the service time at each station of the queueing network to be exponentially distributed, this queueing network model will fall into the class of analytically tractable queueing network models cited in Section 2 where queue dependent service rate at each server is allowed.

7. THE INTERFERENCE PROBLEM IN A LOOSELY COUPLED MULTIPROCESSOR SYSTEM

A computer system is referred to as a loosely coupled multiprocessor system if the processors communicate through secondary storage devices instead of main memory. The model for a tightly coupled multiprocessor system is at level A and that for a loosely coupled multiprocessor system may either be at level B or level C depending upon whether the processors communicate through paging drums or filing disks according to our modelling hierarchy introduced in Section 5. In Figure 7.1 we present a general model for a loosely coupled computer system. Each box indicated by CPU can represent a model of either a uniprocessor or a tightly coupled multiprocessor. In any case, based on the principal of hierarchical modelling, we can replace each of them by an independent service station. The service rate of each CPU is obtained from the lower level model. All secondary storage devices are represented by FCFS service stations.

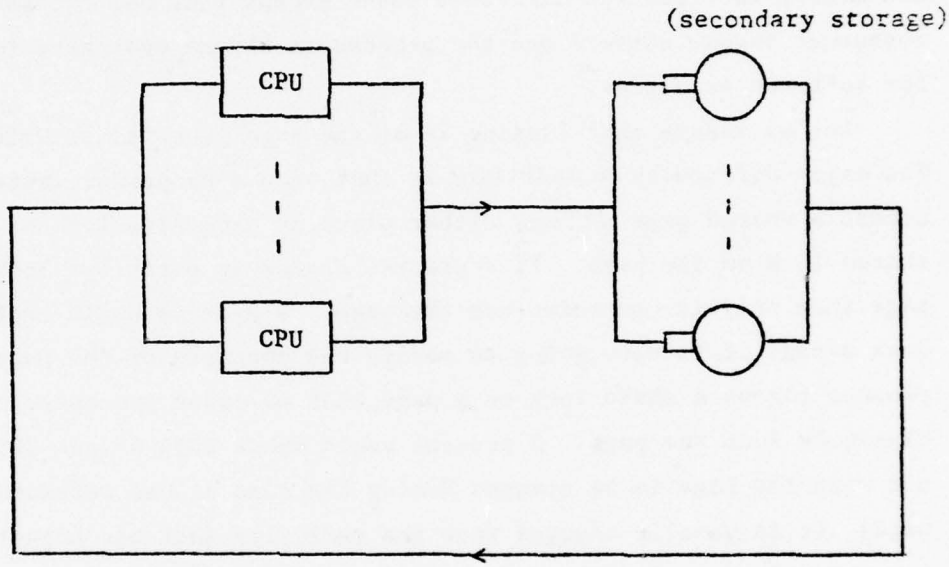


Fig. 7.1 A LOOSELY COUPLED MULTIPROCESSOR SYSTEM MODEL

8. THE SOFTWARE LOCKOUT PROBLEM IN A MULTIPROCESSOR SYSTEM

We now consider the software lockout problem in a multiprocessor system. The software locks are the locks associated with data base systems where elements of the file system are locked by a process to insure meaningful processing by itself and the other processes. The lockout problem is a direct consequence of multiple processors' attempt to process common data bases asynchronously. The situation resembles the memory interference discussed above except that now the software resources become servers and the processors become customers to contend for software resources.

Let us assume that locking is at the page level as in Welch [76]. The major difficulty in modelling is that when a processor wants to access a shared page it may either place an exclusive lock or a shared lock on the page. If a process places an exclusive lock on a page then only it can reference the page. A process would exclusively lock a page if it were going to modify the contents of the page. If a process places a share lock on a page, then no other processes can exclusively lock the page. A process would share lock a page if it did not wish the page to be changed during the time it was referencing the page. It is usually assumed that the exclusive lock has higher priority than the shared lock. Figure 8.1 presents a diagram of the model. To be more specific, a request on a shared page will get access to the page immediately if that page is not exclusively locked and have to wait until there are no pending requests for exclusive lock otherwise. A request on exclusive lock will get immediate access to the page only if the page is unlocked and have to wait until the requests in service and all other currently pending requests on exclusive lock being served otherwise. When a page is unlocked, it always chooses, on first come first served basis, a request on exclusive lock to serve, if any. Otherwise, it will serve all the requests on shared lock similtaneously, if any.

The fact that each shared page can be put under either exclusive lock or shared lock complicates the service discipline of the service

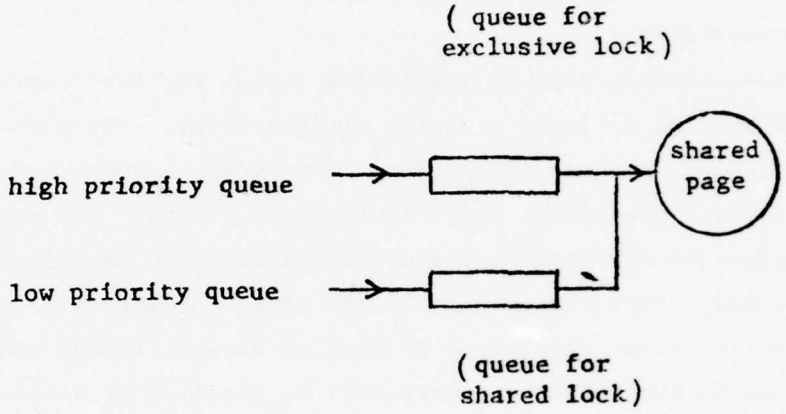


Fig. 8.1 A MODEL FOR A SHARED PAGE WITH TWO KINDS OF LOCKS: EXCLUSIVE LOCK AND SHARED LOCK.

center and makes it different from the conventional types of service centers cited in Section 2. Various authors, e.g. Madnick [58], McCredie [59], Kobayashi [51], Welch [76] etc. have considered the software lockout problem. Almost all the models only allow one kind of lock - exclusive lock. Welch [76] divided the shared pages into two categories, one for exclusively lock only and the other for shared lock only, and treated them separately using M/M/1 and M/M/∞ queueing models, respectively.

We now consider a rough stochestic model for the software lockout problem using a closed queueing network model. The shared pages will be divided into two categories. The first category consists of the pages which are almost never exclusively locked. The second category consists of the rest of the shared pages which are often exclusively locked. Each shared page in the first category will be modelled by an infinite server station as if they were never locked and each shared page in the second category will be modelled by a first come first served station as if they were always exclusively locked. Although the assumption is not quite true, it can be considered as a first order approximation which leads to mathematical tractability. Under the above assumption, the closed queueing model in Figure 8.2 is proposed. Since there is no queueing problem when a processor is not referencing pages in the data base, this stage is in fact an infinite server station.

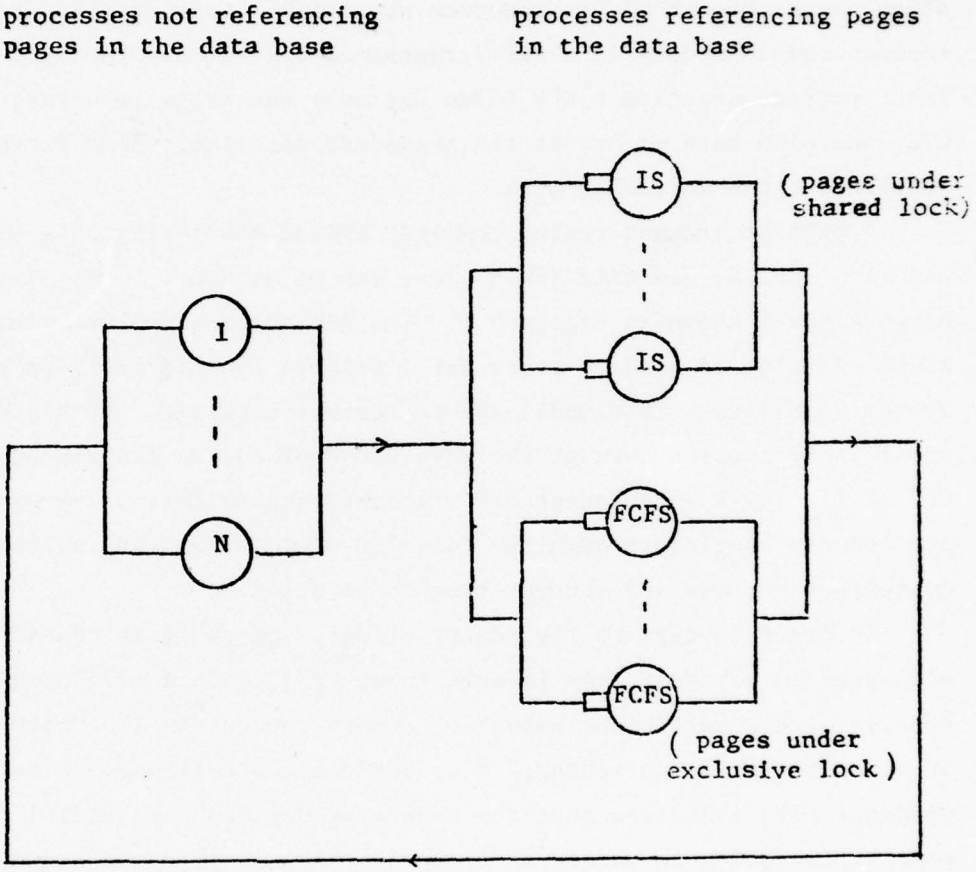


Fig. 8.2 A SIMPLIFIED SOFTWARE LOCKOUT MODEL

9. MODELS OF MULTIPROGRAMMED TIME SHARED COMPUTER SYSTEMS

As pointed out at the beginning of this chapter, one difficulty encountered in modelling computer system is that tasks or processes often occupy more than one resource at a time. Consider the problem encountered in modelling a multiprogrammed paging computer system. Each process executing a CPU holds not only the prime resource, i.e., CPU, but also main memory as the secondary resource. This further complicates the modelling work.

A multiprogrammed paging computer system consisting of a CPU, page I/O devices and file I/O devices can be modelled by the queueing network model shown in Figure 9.1. The CPU station appearing in the model can also be used to represent a tightly coupled multiprocessor system as before. This model can be further extended. If k processors are loosely coupled through the page I/O devices, we can replace the CPU station by k independent CPU stations in parallel. If h processors are loosely coupled through the file I/O devices, we can replicate the dotted box h times and connect them in parallel.

In order to capture the memory effect, let us first consider general program behavior. As is well known [23], each active process must be allocated a sufficient amount of memory to contain its "working set" in order to avoid "thrashing," i.e. performance collapse. Experiment evidence [23] indicates that the mean time between page faults for a process executing in memory space m has the general shape shown in Figure 9.2. This function is referred to as the life time function L(m). Chamberlin, Fuller and Liu [14] proposed a two parameter fit for the life time function.

$$L(m) = \frac{2S}{1+(\frac{d}{m})}$$

where d is the number of pages that provides the process with half of its largest possible life time and S is the expected execution time between page faults when the process is allocated memory space d. Assume that total memory available is of size M. We can also express

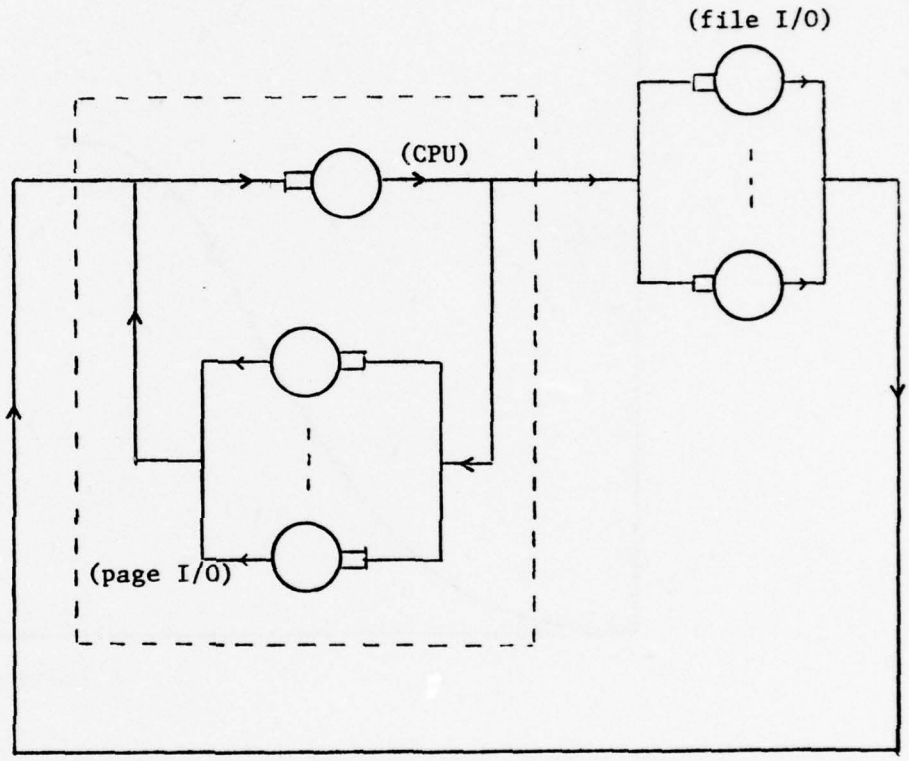


Fig. 9.1 A MULTIPROGRAMMED COMPUTER SYSTEM MODEL

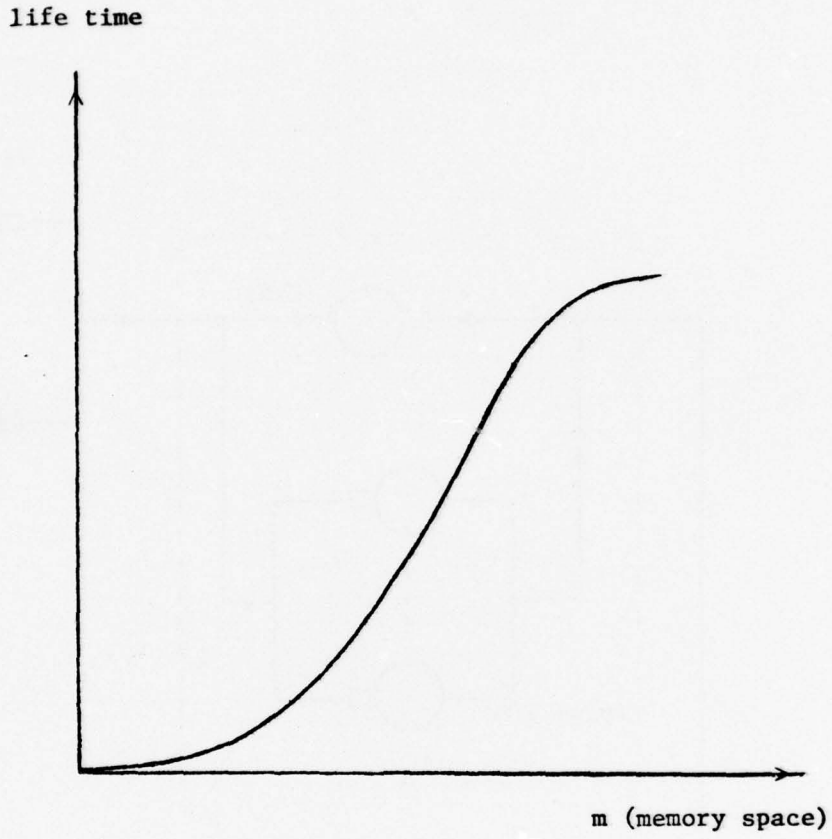


Fig. 9.2 THE LIFE TIME FUNCTION

the life time function in terms of n, the number of processes in memory. Denoting the function by $L^*(n)$, we get

$$L^*(n) = L(\frac{M}{n})$$

under the assumption that memory is equally shared among all programs in memory.

We now return to the queueing model. Let γ be the mean execution time between file I/O requests of a process which is independent of the degree of multiprogramming. Furthermore, let t_1 and t_2 be the CPU overheads on handling a page fault and a file I/O request respectively. These overheads can not be neglected in general. The service completion rate of the CPU can be approximated by $(L^*(n)+t_1)^{-1}+(\mu+t_2)^{-1}$ i.e. the sum of the page fault rate and the file I/O request rate. If the CPU is in fact a tightly coupled multiprocessor, we need to pre-multiply the service completion rate by the CPU utilizations obtained by the memory interference model and the software lockout model to account for the performance degradation due to interactions between multiple processors.

When a file I/O request occurs, if we retain the task in main memory, the number of processes in memory will be fixed, so does the service rate of the CPU. Hence, the queueing network model will fit into the class of models defined in Section 2. However, a more realistic case is to swap the process requesting a file I/O operation out of memory. In this case the number of processes in main memory is equal to the number of processes in the CPU station and page I/O devices. That is to say the service completion rate of the CPU depends on the total number of processes in the two stations, not on the number of processes in itself alone. This kind of situation does not fit into the framework of the queueing network models introduced in Section 2. Nevertheless, the problem can be solved by applying hierarchical modelling. Using the fact that the mean time between page faults is usually an order of magnitude smaller than the mean time between file I/O's, i.e. the interaction between the CPU and page I/O devices is much stronger than their interactions to the file I/O devices, we can consider the CPU and page I/O devices separately from the file I/O

devices. A closed queueing network model, refered to as CPU-PGU model, in Figure 9.3 is proposed to model the interactions between the CPU and page I/O devices. For a given N, the number of processes in main memory, the service completion rates of the CPU and page I/O devices are fixed. So the model fit into the class of models described in Section 2. We can obtain CPU utilization for various values of N, which will be needed in the next level model. Then, going up one level of the modelling hierarchy, we can use the closed queueing network model in Figure 9.4 to model the overall system. Now the subsystem which consists of the CPU and page I/O devices is modeled by a single server whose service completion rate depending upon the number of processes at this service center is equal to the CPU utilization, which we obtain in the lower level model, divided by the mean execution time between file I/O requests of a process. Some numerical results on similar models can be found in [9].

We may further consider a queueing model for an interactive time shared computer system. In this system, we need to model not only the multiprogrammed characteristic of the system, but also the terminal behavior and the job scheduling policy of the operating system. It is well known that if we plot the processor utilization obtained by the multiprogrammed model in Figure 9.4, the curve should take the shape of the dotted curve in Figure 9.5. [20] [5] Note that the dotted curve and the solid curve coincide for $N < N_{max}$. One of the major responsibility of the job scheduling algorithm is to make sure that the memory is not overcommited. That is to say the job scheduling routine will impose on upper bound N_{max} on the number of jobs being activated in main memory. Let N_T be the number of terminals in the system. A closed queueing model in Kobayashi [51] is presented in Figure 9.6.a. As we can see the multiprogrammed computer system model appears as a submodel of the system. The external "job scheduling" queue controlled by the scheduling routine makes the models deviate from conventional queueing model. Since the multiprogrammed computer system model is one step lower in the modelling hierarchy, we can separate it from the rest of the system and model it independently as

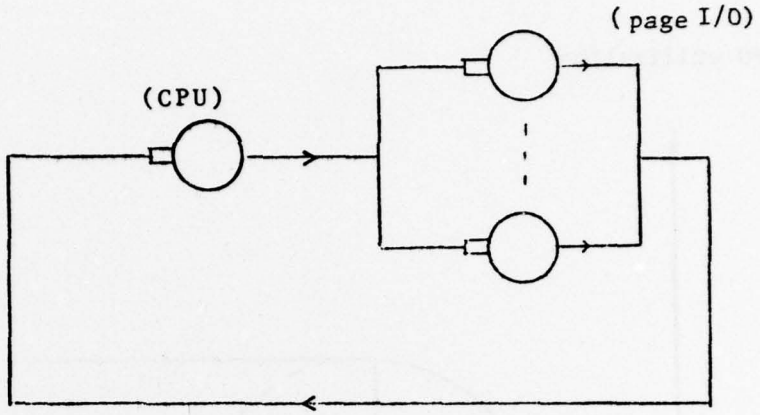


Fig 9.3 A CPU-PGU MODEL

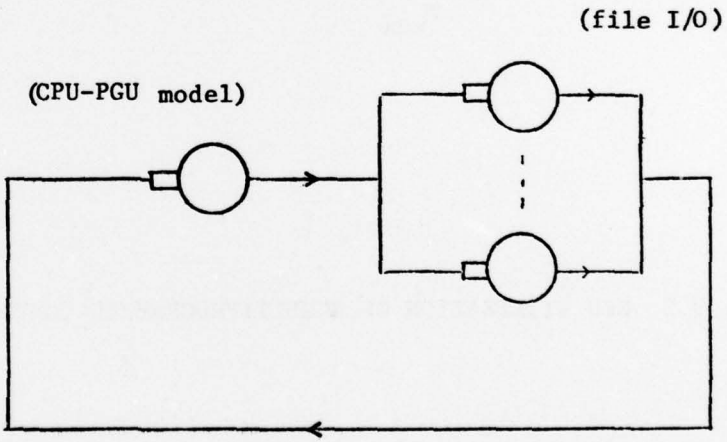


Fig. 9.4 A SIMPLIFIED MULTIPROGRAMMED COMPUTER SYSTEM MODEL

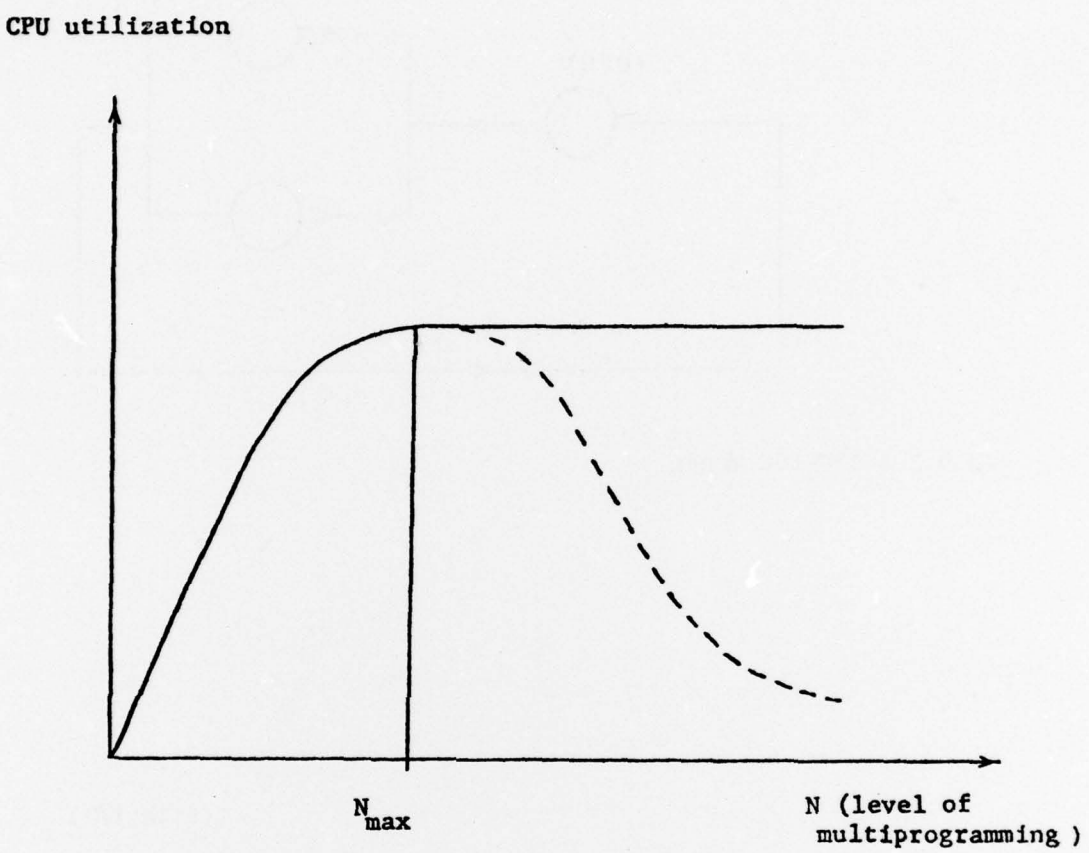
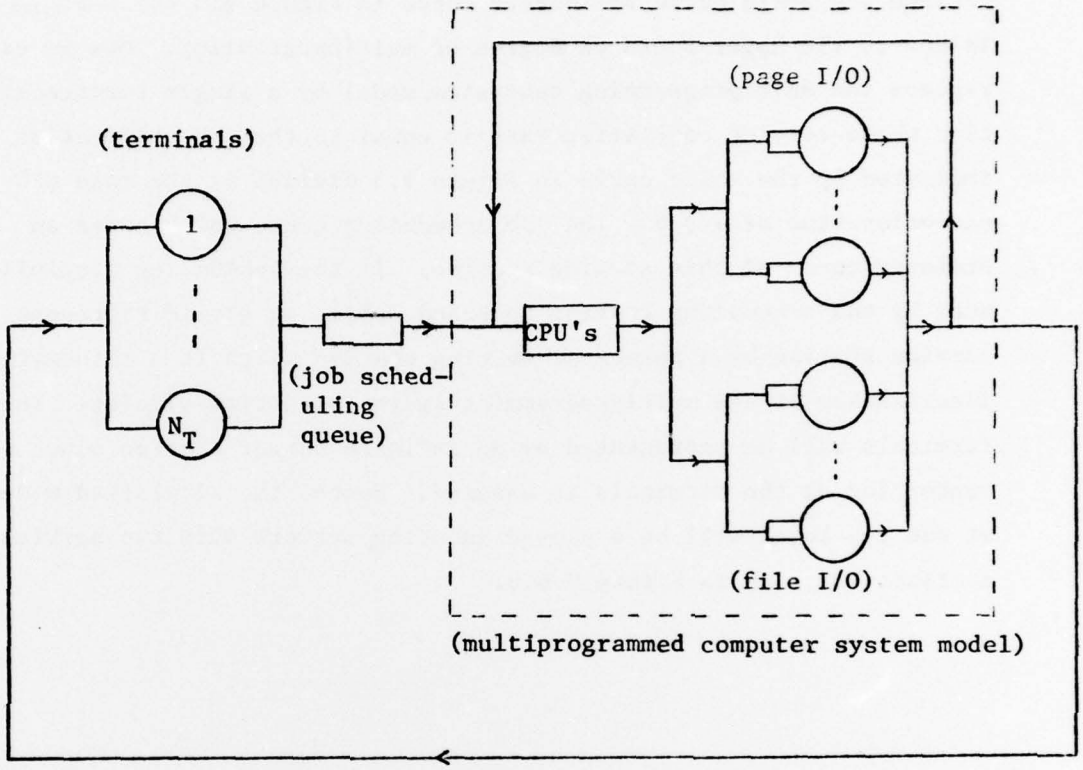
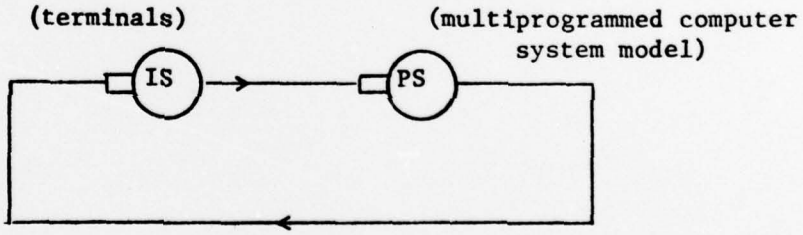


Fig. 9.5 CPU UTILIZATION OF A MULTIPROGRAMMED COMPUTER SYSTEM



(a)



(b)

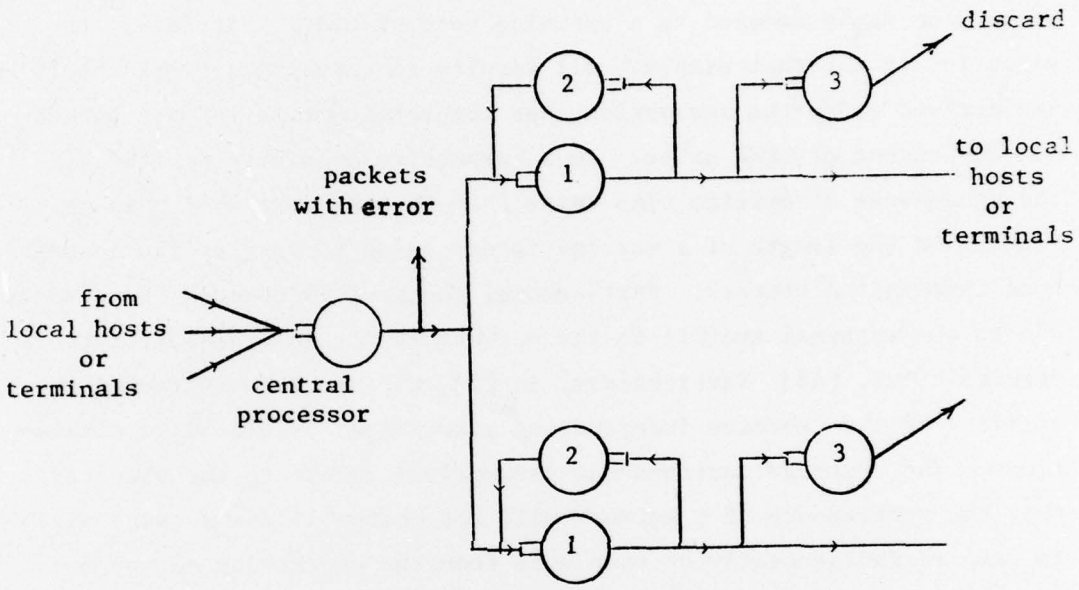
Fig. 9.6 AN INTERACTIVE TIME SHARING SYSTEM MODEL

before except with a constraint on the maximum degree of multiprogramming imposed by the job scheduling routine. The CPU utilization is now given by the solid curve in Figure 9.5. [20] [51] The difference between the solid curve and dotted curve in Figure 9.5 for $N > N_{max}$ is due to the upper bound on degree of multiprogramming. Now we can replace the multiprogramming subsystem model by a single service station whose service completion rate is equal to the CPU utilization indicated by the solid curve in Figure 9.5 divided by the mean CPU execution time of a job. The job scheduling queue now becomes an ordinary queue of this service station. If the scheduling discipline used by the scheduling routine is round robin, we should represent the service station by a processor sharing station which is a mathematical idealization of the multiprogramming system with time slicing. The N_T terminals will be represented by an infinite server station since no contention at the terminals is assumed. Hence, the simplified model at the top level will be a closed queueing network with two service stations as shown in Figure 9.6.b.

10. MODELS OF STORE AND FORWARD COMPUTER COMMUNICATION NETWORKS VIA TERRESTRIAL LINKS

Modelling of store and forward computer communication networks via terrestrial links is primarily due to Kleinrock [47, 48]. If we view channels as servers and messages as customers or tasks, the problem seems to be again reduced to a queueing network model. Actually, the situation is not that simple. All results on queueing network models are derived under the assumption that the service time at each server is independent of each other. In a computer communication network, the randomness of service time comes from the randomness of message length and the length of a message is not going to vary as the message hops through the network. Furthermore, the arrival process of messages due to the internal traffic in the network is not independent of their service times. [48] Nevertheless, in [2], Kleinrock investigates the validity of the "message independence assumption" by extensive simulations. The "message independence assumption" refers to the property that the performance of a network will not change if the message size is sampled independently at each node from the underlying common message size distribution instead of keeping the message size fixed when transmitting through the network. The simulation results indicate that under most conditions of interest this assumption is reasonable. The message independent assumption provides us with the necessary foundation for applying the queueing network models to analyze store and forward computer communication networks via terrestrial links.

Nevertheless, an infinite storage capacity at each store and forward node has been implicitly assumed in the model. Lam [56] relaxed this assumption and presented a queueing network model for each store and forward node to incorporate the constraints of finite nodal storage capacities, as well as the channel transmission control mechanisms of positive acknowledgement and time-out of packets. This model is presented in Figure 10.1. Finally these single node results are interfaced by imposing a continuity of flow constraint. Again we see, depending upon the levels of details required, queueing models of different complexities may be employed.



1 = output processor (FCFS)

2 = time-out delay (IS)

3 = acknowledgement delay (IS)

Fig. 10.1 THE NODAL MODEL OF A COMPUTER COMMUNICATION NETWORK [56]

11. REMARK

While analytic results are clearly not powerful enough to provide a "cookbook" approach to performance analysis and prediction, general methodology and difficulties on model formulation are discussed through examinations of various computer system and network models. These models are presented in a systematic way based on the powerful modelling methodology - hierarchical modelling.

BIBLIOGRAPHY

1. Anderson, H.A. and Sargent, R. "The Statistical Evaluation of the Performance of an Experimental APL/360 System," Statistical Computer Performance Evaluation, W. Freiberger (ed.) Academic Press, 1972, pp. 73-98.

2. Ando, A. and Fisher, F.M. "Near Decomposability, Partition and Aggregation, and the Relevance of Stability Discussions" International Economic Review 4, Jan. 1963, pp. 53-67.

3. Baskett, F., "The Dependence of Computer System Queues Upon Processing Time Distribution and Central Processor Scheduling," Proc. of the ACM SIGOPS Third Symposium on Operating System Principles, 1971, pp. 109-113.

4. Baskett, F., Chandy, K.M., Muntz, R.R. and Palacios, F.G., "Open, Closed and Mixed Networks of Queues With Different Classes of Customers," J. ACM 22, April 1975, pp. 248-260.

5. Baskett, F., and Muntz, R.R., "Queueing Network Models With Different Classes of Customers," Proc. Sixth Annual IEEE Computer Society Int'l Conf., Sept. 1972, pp. 205-209.

6. Baskett, F. and Palacios, F.G., "Processor Sharing in a Central Server Queueing Model of Multiprogramming With Applications," Proc. of the Sixth Annual Princeton Conference on Information Sciences and Systems, March 1972.

7. Belady, L., and Kuehner, C.J., "Dynamic Space Sharing in Computer Systems," Comm. ACM 12, May 1969, pp. 282-288.

8. Bhandarkar, D.P., "Analysis of Memory Interference in Multiprocessors," IEEE Trans. Compt., C-24, Sept. 1975, pp. 897-907.

9. Brandwajn, A., "A Model of a Virtual Memory System," Acta Informatica 6, 1976, pp. 365-386.

10. Brown, R.M., Browne, J.C. and Chandy, K.M., "Memory Management and Response Time," Comm. ACM 20, March 1977, pp. 153-165.

11. Browne, J.C., et. al., "Hierarchical Techniques for the Development of Realistic Models of Complex Computer Systems," Proc. IEEE 63, June 1975, pp. 996-975.

12. Buzen, J., "Queueing Network Models of Multiprogramming," Ph.D. Thesis, Harvard University, 1971.

13. Buzen, J., "Computational Algorithms for Closed Queueing Networks With Exponential Servers," Comm. ACM 16, Sept. 1973, pp. 527-531.

14. Chamberlin, D., Fuller, S. and Liu, L.Y., "An Analysis of Page Allocation Strategies for Multiprogramming Systems With Virtual Memory," IBM J. Res. Develop. 17, 1973, pp. 404-412.

15. Chandy, K.M., "The Analysis and Solutions for General Queueing Networks," Proc. of the Sixth Annual Princeton Conference on Information Sciences and Systems, March 1972, pp. 224-228.

16. Chandy, K.M., Howard, J.H. and Towsley, D.F., "Product Form and Local Balance in Queueing Networks," J. ACM 24, April 1977, pp. 250-263.

17. Chandy, K.M., Keller, T.W. and Browne, J.C., "Design Automation and Queueing Networks: An Interactive System for the Evaluation of Computer Queueing Models," Proc. of the Ninth Annual Design Automation Workshop, June 1972.

18. Chow, W.M., "A Memory Interference Model for Multiprocessor Systems," IBM Res. Report RC4867, 1974.

19. Courtois, P.J., "On the Near-Complete-Decomposability of Networks of Queues and of Stochastic Models of Multiprogramming Computer Systems," Rep. CMU-CS-72-11, Carnegie-Mellon U., 1971.

20. Courtois, P.J., "Decomposability, Instabilities and Saturation in Multiprogramming Systems," Comm. ACM 18, July 1975, pp. 371-377.

21. Crane, M.A., and Iglehart, D.L.,"Simulating Stable Stochastic System, I: General Multiserver Queues," J. ACM 21, Jan. 1974, pp. 103-113.

22. Cox, D.R., "A Use of Complex Probabilities in Theory of Stochastic Processes," Proc. Cambridge Phil. Soc. 51, 1955, pp. 313-319.

23. Denning, P.J. and Graham, G.S., "Multiprogrammed Memory Management," Proc. IEEE 63, June 1975, pp. 924-949.

24. Feller, W., "Diffusion Process in One Dimension," Trans. Math. Soc. 77, 1954, pp. 1-31.

25. Feller, W., An Introduction to Probability Theory and Its Application, Vol. 1, 3rd ed., Wiley, New York, 1968.

26. Ferdinand, A.E., "An Analysis of the Machine Interference Model," IBM Systems J., 10.2, 1971, pp. 129-142.

27. Flynn, M.J., "Some Computer Organizations and Their Effectiveness," IEEE Trans. Compt., C-21, Sept. 1972, pp. 848-860.

28. Franta, W.R., "The Mathematical Analysis of the Computer System Modeled as a Two Stage Cyclic Queue," Acta Informatica 6, 1976 pp. 187-209.

29. Fuller, S.H., and Baskett, F., "An Analysis of Drum Storage Units," J. ACM 22, Jan. 1975, pp. 83-105.

30. Gaver, D.P., "Diffusion Approximations and Models for Certain Congestion Problems," JAP 5, 1968, pp. 607-623.

31. Gaver, D.P., "Analysis of Remote Terminal Backlogs Under Heavy Demand Conditions," J. ACM 18, July 1971, pp. 405-415.

32. Gaver, D.P., and Shedler, G.S., "Processor Utilization in Multiprogramming Systems via Diffusion Approximations," Operation Research 21, 1973, pp. 569-576.

33. Gaver, D.P., and Shedler, G.S., "Approximate Models for Processor Utilization in Multiprogrammed Computer Systems," SIAM J. Computing 2, Sept. 1973, pp. 183-192.

34. Gelenbe, E., "On Approximate Computer System Models," J. ACM 22, April 1975, pp. 261-269.

35. Gelenbe, E., "A Non-Markovian Diffusion Model and Its Application to the Approximation of Queueing System Behavior," TR 158, RIA, March 1976.

36. Gelenbe, E., and Pujolle, G., "The Behavior of a Single Queue in a General Queueing Network," Acta Informatica 7, 1976, pp. 123-136.

37. Gordon, W.J., and Newell, G.F., "Closed Queueing Systems With Exponential Servers," Operation Research, 1967, pp. 254-265.

38. Halachmi, B., and Franta, W.R., "A Closed, Cyclic, Two Stage Multiprogrammed System Model and Its Diffusion Approximation Solution," Proc. SIGMETRIC Symposium, Montreal, October 1974; ACM Performance Evaluation Review 3, 1974, pp. 54-64.

39. Halachmi, B., and Franta, W.R., "A Diffusion Approximate Solution to the G/G/K Queueing System," TR-75-3, Department of Computer Science, the University of Kansas.

40. Iglehart, D.L., "Queueing Theory," preprint, Operation Research Dept., Stanford University.

41. Iglehart, D.L., and Shedler, G.S., "Estimation via Regenerative Simulation of Response Times in Networks of Queues," IBM Res. Report RJ1740, 1976. (to appear in J. ACM)

42. Jackson, J.R., "Jobshop-Like Queueing Systems," Management Science 10, 1, Oct. 1963, pp. 131-142.

43. Karlin, S. and Taylor, H.M., A First Course in Stochastic Processes, 2nd edition, Academic Press, New York, 1975.

44. Keilson, J., "The Role of Green's Functions in Congestion Theory," Proceedings of the Symposium on Congestion Theory, W.L. Smith and W.E. Wilkinson (ed)., University of North Carolina, 1965, pp. 43-71.

45. Kingman, J.F.C., "Some Inequalities for the Queue in the GI/G/1," Biometrika 49, 1962, pp. 315-324.

46. Kingman, J.F.C., "The Heavy Traffic Approximation in the Theory of Queues," Proceedings of the Symposium on Congestion Theory, W.L. Smith and W.E. Wilkinson (ed), U. of North Carolina Press, 1965, pp. 137-169.

47. Kleinrock, L., "Analytic and Simulation Models in Computer Network Design," Spring Joint Computer Conference, AFIPS Conference Proceedings, vol. 36, 1970, pp. 569-579.

48. Kleinrock, L., Communication Nets, McGraw-Hill, New York, 1964, reprinted by Dover Publications, 1972

49. Kleinrock, L., Queueing Systems, vol. 1: Theory, vol. 2: Computer Applications," Wiley Interscience, New York 1975.

50. Kobayashi, H., "Application of the Diffusion Approximation to Queueing Networks, I, II," J. ACM 21, 1974, pp. 316-328, 459-469.

51. Kobayashi, H., "System Design and Performance Analysis Using Analytic Models," IBM Res. Report RA75, 1975.

52. Kobayashi, H., and Konheim, A.G., "Queueing Models for Computer Communications System Analysis," IEEE Trans. Comm., Com-25, January 1977, pp. 2-28.

53. Kobayashi, H., Onozato, Y., and Huynch, D., "An Approximate Method for Design and Analysis of an ALOHA System," IEEE Trans. Commun., Com-25, January 1977, pp. 148-157.

54. Kobayashi, H. and Reiser, M., "On Generalization of Job Routing Behavior in a Queueing Network Model," IBM Res. Report RC5252, 1975.

55. Lam, S.S., "Queueing Networks with Lost and Triggered Arrivals," IBM Res. Report RC5540, 1975.

56. Lam, S.S., "Store and Forward Buffer Requirement in a Packet Switching Network," IEEE Trans. Commun., Com-24, April 1976, pp. 394-403.

57. Lewis, P.A., and Shedler, G.S., "Empirically Derived Micromodels for Sequences of Page Exceptions," IBM J. Res. Develop. 17, March 1973.

58. Madnick, S.E., "Multiprocessor Software Lockout," Proc. 1968 ACM, Nat. Conf, pp. 19-24.

59. McCredie, J., "Analytic Models as Aids in Multiprocessor Design," Ph.D. dissertation, Carnegie Mellon U., 1972.

60. Moore, C.G. III, "Network Models for Large Scale Time Sharing Systems," TR 71-1, Dep. of I.E., U. of Michigan at Ann Arbor, 1971.

61. Muntz, R.R. and Baskett, F., "Open, Closed and Mixed Networks of Queues With Different Classes of Customers," TR 23, Digital System Lab., Stanford University, 1972.

62. Newell, G.F., Applications of Queueing Theory, 1971, Chapman and Hall, Ltd., London, 1971.

63. Palacios, F.G., "An Analytic Model of a Multiprogramming System Including a Job Mix," TR 4, Dept. of C.S., U. of Texas at Austin, 1972.

64. Ponzin, L., "CIGALE, the Packet-Switching Machine of the Cyclades Computer Network," Inf. Proc. 74, Stockholm, North-Holland, August 1974, pp. 155-159.

65. Posner, M. and Bernholtz, B., "Closed Finite Queueing Networks With Time Lags and With Several Classes of Units," Operation Research 16, 1968, pp. 977-985.

66. Price, T., "A Note on the Effect of the Central Processor Service Time Distribution on Processor Utilization in Multiprogrammed Computer Systems," J. ACM 25, April 1976, pp. 342-346.

67. Reiser, M., and Kobayashi, H., "Accuracy of the Diffusion Approximation for Some Queueing Systems," IBM J. Res. Develop. 18, March 1974.

68. Reiser, M., and Kobayashi, H., "Recursive Algorithm for General Queueing Networks With Exponential Servers," IBM Res. Report RC4254, 1973.

69. Reiser, M., and Kobayashi, H., "Queueing Networks With Multiple Closed Chains: Theory and Computational Algorithms," IBM J. Res. Develop. 19, May 1975, pp. 283-294.

70. Sekata, M., Noguchi, S. and Oizumi, J., "Analysis of a Processor Shared Queueing Model for Time-Sharing Systems," Proc. of the Second Hawaii International Conf. on System Sciences, Jan. 1969.

71. Sekino, A., "Performance Evaluation of Multiprogrammed Time-Shared Computer Systems," Ph.D. dissertation, Dept. of E.E., MIT, 1972.

72. Shedler, G.S., "A Cyclic Queue Model of a Paging Machine," IBM Res. Report RC2814, 1970.

73. Simon, H., and Ando, A., "Aggregations of Variables in Dynamic Systems," Econometrica 20, 1961, pp. 111-138.

74. Skinner, C., and Asher, J., "Effect of Storage Contention on System Performance," IBM Systems J. 8, April 1969, pp. 319-333.

75. Tang, C.K., "Cache System Design in the Tightly Coupled Multiprocessor System," AFIPS Conference Proceedings, Vol. 45, 1976, pp. 749-753.

76. Welch, P.D., "On the Interface Between Software and Hardware Models," IBM Res. Report RC5770, 1975.

77. Whittle, P., "Nonlinear Migration Processes," Proc. 36th Session of the International Statistical Institute, 1969, pp. 642-647.

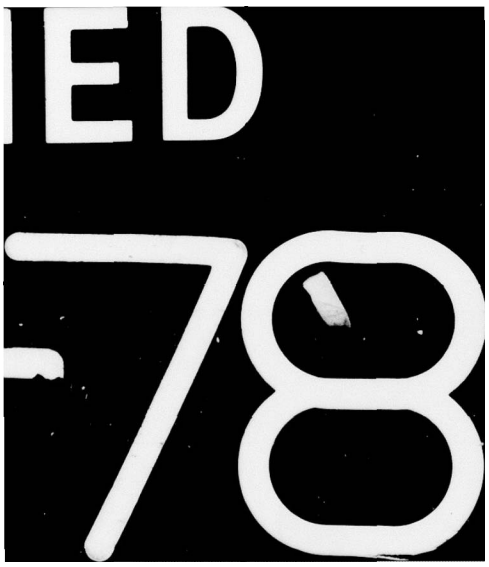
78. Whittle, P., "Equilibrium Distributions for an Open Migration Process," JAP 5, 1968, pp. 567-571.

79. Wilhelm, N.C., "A General Model for the Performance of Disk Systems," J. ACM 24, Jan. 1977, pp. 14-31.

80. Yu, P.S., "On Accuracy Improvement and Applicability Conditions of Diffusion Approximation With Applications to Modelling of Computer Systems," TR 129, Digital System Lab., Stanford U., Jan. 1977.

81. Yu, P.S., "Passage Time Distribution for a Class of Queueing Networks: Closed, Open or Mixed With Different Classes of Customers With Applications to Computer System Modelling," TR 135, Digital System Lab., Stanford U., March 1977.

82. Yu, P.S., "Performance Analysis of Computer Communication Networks via Random Access Channels," TR 137, Digital System Lab., Stanford U., April 1977.


ED
78